МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Директор ГБУЗ «ЧОМИАЦ»

_____ А.С. Староверов

«___»_____ 2024 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка оболочки для автоматизации выполнения интеллектуального анализа временных рядов

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.04.04.2024.308-1496.ВКР

Научный руководитель,
профессор кафедры СП, д.ф.-м.н.,
доцент
_____ М.Л. Цымблер

Автор работы,
студент группы КЭ-229
_____ М.С. Подседов

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы магистранта

студенту группы КЭ-229
Подседову Максиму Сергеевичу,
обучающемуся по направлению
09.04.04 «Программная инженерия» (магистерская программа
«Искусственный интеллект и инженерия данных»)

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. №764-13/12)
Разработка оболочки для автоматизации выполнения интеллектуального анализа временных рядов.
2. **Срок сдачи студентом законченной работы:** 20.05.2024 г.
3. **Исходные данные к работе**

3.1. Esling P., Agon C. Time-series data mining. // ACM Comput. Surv., 2012. – Vol. 45, no. 1. – P. 12:1–12:34.

3.2. Иванова Е.В., Цымблер М.Л. Внедрение концепции матричного профиля в реляционную СУБД для интеллектуального анализа временных рядов. // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика, 2021. – Т. 10, № 3. – С. 72–87.

4. **Перечень подлежащих разработке вопросов**

4.1. Выполнить анализ предметной области и разработать спецификацию требований к оболочке.

4.2. Выполнить проектирование оболочки, включая веб-интерфейс пользователя, структуру базы данных и модульную структуру системы.

4.3. Выполнить реализацию, тестирование и отладку оболочки.

4.4. Выполнить эксперименты по исследованию эффективности работы оболочки.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
профессор кафедры СП, д.ф.-м.н., доцент М.Л. Цымблер

Задание принял к исполнению М.С. Подседов

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Интеллектуальный анализ данных (Data Mining) представляет собой совокупность методов и алгоритмов для обнаружения в данных ранее неизвестных и практически полезных знаний, используемых для принятия стратегически важных решений в различных сферах человеческой деятельности [21]. Временные ряды являются одним из наиболее важных классов данных, подвергаемых интеллектуальному анализу. Под временным рядом (time series) понимают последовательность хронологически упорядоченных вещественных значений.

Развитие современных технологий и рост объемов данных в последние десятилетия привели к возникновению новых требований к обработке и анализу информации. Особенно важным стал анализ временных рядов, который находит применение в различных областях, таких как финансовые рынки, метеорология, медицина и многие другие.

Актуальность

В настоящее время инструменты для анализа временных рядов могут быть сложными в использовании, требовать специальных навыков и занимать много времени на реализацию. Кроме того, на текущий момент отсутствует универсальный инструмент, объединяющий возможности хранения данных, автоматизированного интеллектуального анализа временных рядов и их визуализации. Разработанная оболочка для автоматизации выполнения интеллектуального анализа временных рядов представляет собой решение указанных проблем, обеспечивая сокращение времени, потраченного на анализ временных рядов, и упрощение процесса работы для пользователей.

Постановка задачи

Целью данной выпускной квалификационной работы является разработка оболочки для автоматизации выполнения интеллектуального анализа временных рядов.

Для достижения поставленной цели необходимо решить следующие задачи:

1) выполнить анализ предметной области и разработать спецификацию требований к оболочке;

2) выполнить проектирование оболочки, включая веб-интерфейс пользователя, структуру базы данных и модульную структуру системы;

3) выполнить реализацию, тестирование и отладку оболочки;

4) выполнить эксперименты по исследованию эффективности работы оболочки.

Структура и содержание работы

Работа состоит из введения, 4 глав, заключения и списка используемой литературы. Объем работы составляет 49 страниц, объем библиографии – 31 наименование.

Первый раздел «Современные методы интеллектуального анализа временных рядов» содержит формальные определения и обзор существующих решений для анализа временных рядов.

Второй раздел «Проектирование» содержит спецификацию требований к оболочке, архитектуру системы, варианты использования оболочки и схему базы данных.

Третий раздел «Реализация» содержит технические аспекты реализации отдельных компонентов оболочки.

Четвертый раздел «Тестирование и эксперименты» содержит результаты функционального тестирования и исследования скорости выполнения функции поиска примитивов

Заключение резюмирует результаты, полученные в рамках исследования.

В приложениях представлены структура таблиц базы данных и макеты пользовательского интерфейса.

1. СОВРЕМЕННЫЕ МЕТОДЫ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ВРЕМЕННЫХ РЯДОВ

1.1. Теоретический базис

В данном разделе приводятся нотация и формализованные определения.

Временной ряд представляет собой хронологически упорядоченную последовательность числовых значений, формальное определение представлено в формуле (1):

$$T = (t_1, \ldots, t_n), t_i \in \mathbb{R}. \quad (1)$$

Подпоследовательность $T_{i,m}$ временного ряда T представляет собой непрерывное подмножество T, состоящее из m элементов и начинающееся с позиции i, формальное определение представлено в формуле (2):

$$T_{i,m} = (t_i, \ldots, t_{i+m-1}), \quad (2)$$

где $1 \leq i \leq n - m + 1, 1 \leq m \ll n$.

Множество всех подпоследовательностей S_T^A временного ряда T является упорядоченным набором всех возможных подпоследовательностей длины m, содержащихся в T. Подпоследовательности сортируются в порядке возрастания индекса их первого элемента. В формуле (3) представлено формальное определение множества всех подпоследовательностей:

$$S_T^A = \{T_{1,m}, T_{2,m}, \ldots, T_{n-m+1,m}\}. \quad (3)$$

Профиль расстояния D_i^T вычисляется по формуле (4) и представляет собой вектор расстояний между заданной подпоследовательностью $T_{i,m}$ и каждой подпоследовательностью $T_{j,m}$ из множества всех подпоследовательностей:

$$D_i^T = \left(dist\left(T_{i,m}, T_{1,m}\right), \ldots, dist\left(T_{i,m}, T_{n-m+1,m}\right)\right), \quad (4)$$

где $dist(,)$ означает евклидово расстояние между нормализованными подпоследовательностями.

С помощью профиля расстояния выполняется поиск ближайшего соседа для каждой подпоследовательности временного ряда, за исключением

тривиальных соседей. По формуле (5) определяется ближайший сосед. Подпоследовательность $T_{j,m}$ называется ближайшим соседом подпоследовательности $T_{i,m}$:

$$dist(T_{i,m}, T_{j,m}) = min(D_i^T). \quad (5)$$

Подпоследовательность $T_{j,m}$ является тривиальным соседом для $T_{i,m}$, если $|i - j| \leq \frac{m}{2}$.

Пусть неотрицательная симметричная функция используется в качестве функции расстояния, она представлена в формуле (6):

$$dist: \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}\,. \quad (6)$$

Матричный профиль (МП) P^T временного ряда T представляет собой вектор расстояний между каждой подпоследовательностью $T_{i,m}$ и ее ближайшим нетривиальным соседом, формальное определение представлено в формуле (7):

$$P^T = \left(nn_{nt}(D_1^T), \dots, nn_{nt}(D_{n-m+1}^T)\right), \quad (7)$$

где $nn_{nt}(D_i^T)$ означает минимальное расстояние между $T_{i,m}$ и ее нетривиальными соседями [27,30].

С МП ассоциируется дополнительная структура данных для определения местоположения ближайших нетривиальных соседей. Индекс МП профиля I^T вычисляется по формуле (8) временного ряда T представляет собой вектор, хранящий индекс ближайшего нетривиального соседа для каждой подпоследовательности:

$$I^T = (I_1^T, \dots, I_{n-m+1}^T), \quad (8)$$

где $I_i^T = j$, если $nn_{nt}(D_i^T) = dist(T_{i,m}, T_{j,m})$.

МП представляет собой агностический (не зависящий от предметной области) инструмент интеллектуального анализа временного ряда, в котором исследователь должен указать лишь один параметр – длину подпоследовательности. Его можно рассматривать как метаданные для аннотирования соответствующего временного ряда. Например, максимальное значение профиля соответствует аномальной подпоследовательности (называе-

мой диссонансом), тогда как минимальные значения соответствуют наиболее похожей паре подпоследовательностей в ряде (называемой мотивом).

Есть несколько алгоритмов для поиска МП. Одним из них является алгоритм STAMP (Space-Time Matrix Profile), предложенный в работе [26] предназначен для нахождения повторяющихся подпоследовательностей в временных рядах. Основная идея заключается в поиске схожих фрагментов в ряде, которые могут представлять повторения или аномалии.

В работе [29] представлена улучшенная версия алгоритма STAMP под названием STOMP (Scalable-Time-Ordered Matrix Profile), которая решает проблему масштабирования для больших временных рядов. В алгоритме STOMP используется упрощенное представление матрицы профиля, которое позволяет снизить вычислительную сложность и увеличить производительность.

Как было сказано выше, с помощью МП можно найти мотив (motif), который был предложен в работе [17]. Поиск мотивов (шаблонов) предполагает нахождение пар непересекающихся подпоследовательностей временного ряда, наиболее похожих друг на друга и формально определяется следующим. Формальное определение мотива представлено в формуле (9). Пара подпоследовательностей $\{T_{i,m}, T_{j,m}\}$ ряда T называется мотивом, если они имеют наибольшую схожесть по сравнению с другими парами подпоследовательностей $(T_{a,m}, T_{b,m})$ этого временного ряда:

$$\begin{gathered} \forall a, b, |a-b| \geq \text{ש}, i, j, |i-j| \geq \text{ש}, \text{ש} > 0\ , \\ dist(T_{i,m}, T_{j,m}) \leq dist(T_{a,m}, T_{b,m}), \end{gathered} \tag{9}$$

где ש – параметр, определяющий минимальный промежуток, на который должны отстоять друг от друга подпоследовательности в мотиве.

Данный параметр позволяет отбрасывать мотивы, которые состоят из взаимопересекающихся подпоследовательностей и потому не имеют практической ценности.

Алгоритм MK, предложенный в той же работе, что и мотив, предназначен для точного нахождения мотива во временном ряде. Он основан на следующем принципе: каждую подпоследовательность исходного ряда можно рассматривать как конечное множество точек в метрическом пространстве. Путем случайного выбора опорной точки из этого множества и упорядочивания точек исходного множества по возрастанию расстояния до опорной точки можно получить линейный порядок. Для любой пары точек, не являющихся опорной, можно утверждать, что если они близки друг к другу в исходном пространстве, то они будут находиться близко и в линейном порядке. Это свойство в сочетании с неравенством треугольника позволяет отсеивать подпоследовательности, которые не могут быть мотивом, без необходимости вычисления расстояний. Для более эффективного сужения пространства поиска алгоритм использует несколько опорных точек.

Поиск аномалий предполагает обнаружение подпоследовательностей временного ряда, которые наиболее непохожи на все остальные подпоследовательности ряда. Концепция диссонанса (discord), предложенная в работе [14], уточняет и формализует понятие аномалии и в настоящее время признается научным сообществом как наиболее адекватный способ поиска аномалий во временном ряде. Диссонанс определяется следующим образом. Подпоследовательности $T_{i,m}$ и $T_{j,m}$ ряда T называются непересекающимися, если$|i-j| \geq m$. Подпоследовательность, которая является непересекающейся к данной подпоследовательности $T_{i,m}$, обозначается как $M_{T_{i,m}}$. Формальное определение диссонанса представлено в формуле (10). Подпоследовательность $T_{i,m}$ является диссонансом:

$$\forall T_{j,m}, M_{T_{i,m}} \epsilon\, T\; min\left(dist\left(T_{i,m}, M_{T_{i,m}}\right)\right) > min\left(dist\left(T_{j,m}, M_{T_{j,m}}\right)\right). \quad (10)$$

Иными словами, диссонанс представляет собой подпоследовательность ряда, имеющую максимальное расстояние до наиболее близкой к ней непересекающейся подпоследовательности.

Одним из алгоритмов поиска диссонанса является алгоритм HOTSAX (Heuristically Ordered Time series using Symbolic Aggregate approXimation) [15], который базируется на преобразовании временного ряда в символьную последовательность (Symbolic Aggregate approXimation, SAX) и последующем выявлении «горячих точек» (hot spots), то есть участков ряда, где происходит наибольшее отклонение от нормы.

В работах [11,31] предложено понятие сниппета (snippet). Поиск сниппетов предполагает нахождение значимых и информативных фрагментов данных, которые отображают основные характеристики временного ряда. Концепция сниппета уточняет понятие типичной подпоследовательности временного ряда следующим образом. Временной ряд T может быть разбит на сегменты заданной длины m, где каждый сегмент S_i представляет собой подпоследовательность ряда, показанной в формуле (11):

$$T_{m\cdot(i-1)+1,m}, \tag{11}$$

где $1 \le i \le \frac{n}{m}$. Здесь и далее без существенного ограничения общности считается, что n кратно m, поскольку $m \ll n$.

Каждый сниппет представляет собой один из сегментов временного ряда. Со сниппетом ассоциируются его ближайшие соседи – подпоследовательности ряда, имеющие ту же длину, что и сниппет, которые более похожи на данный сниппет, чем на другие сегменты. Для вычисления схожести подпоследовательностей используется специализированная мера схожести, основанная на евклидовом расстоянии. Сниппеты упорядочиваются по убыванию мощности множества своих ближайших соседей.

Для вычисления схожести подпоследовательностей при нахождении сниппетов применяется мера MPdist [8], неформально определяется следующим образом. Два временных ряда равной длины m тем более похожи друг на друга в смысле меры MPdist, чем больше в каждом из них имеется подпоследовательностей заданной длины $\ell(3 \le \ell \le m)$, близких друг к другу в смысле нормализованного евклидова расстояния. Мера MPdist

устойчива к выбросам, шумам и пропущенным значениям во временном ряде.

Для нахождения сниппетов используется алгоритм Snippet-Finder предложенный в работе [12], который включает в себя MPdist для поиска и сравнения сниппетов во временных рядах. Основой алгоритма являются методы анализа временных рядов, статистические меры и пороговые значения, которые позволяют определить, является ли сегмент сниппетом или нет.

Левый профиль расстояний DL_i временного ряда T представляет собой вектор евклидовых расстояний между заданной подпоследовательностью $T_{i,m}$ и каждой подпоследовательностью, которая появляется до нее во временном ряде. Формальное определение представлено в формуле (12):

$$DL_i = \left[d_{i,1}, d_{i,2}, \dots, d_{i,i-m/4}\right], \tag{12}$$

где $d_{i,j} = ED_{norm}(T_{i,m}, T_{j,m}) = dist(T_{i,m}, T_{\text{j},m})$.

Правый профиль расстояний DR_i временного ряда T представляет собой вектор евклидовых расстояний между заданной подпоследовательностью $T_{i,m}$ и каждой подпоследовательностью, которая появляется после нее во временном ряде. Формальное определение представлено в формуле (13):

$$DR_i = \left[d_{i,i+m/4}, d_{i,i+m/4+1}, \dots, d_{i,n-m+1}\right]. \tag{13}$$

Можно легко найти левого ближайшего соседа подпоследовательности $T_{i,m}$ из левого профиля расстояний, а правого ближайшего соседа $T_{i,m}$ из правого профиля расстояний.

Ближайший сосед слева $LNN(T_{i,m})$ это подпоследовательность, которая появляется до $T_{i,m}$ во временном ряде и наиболее похожа на нее. Формальное определение представлено в формуле (14):

$$LNN(T_{i,m}) = T_{j,m}, \tag{14}$$

если $d_{i,j} = min(DL_i)$.

Ближайший сосед слева $RNN(T_{i,m})$ это подпоследовательность, которая появляется после $T_{i,m}$ во временном ряде и наиболее похожа на нее. Формальное определение представлено в формуле (15):

$$RNN(T_{i,m}) = T_{j,m}, \quad (15)$$

если $d_{i,j} = min(DR_i)$.

Левый МП PL временного ряда T представляет собой вектор z-нормированного евклидова расстояния между каждой подпоследовательностью $T_{i,m}$ и ее левым ближайшим соседом во временном ряду. Формальное определение представлено в формуле (16):

$$PL = [min(DL_1), min(DL_2), \dots, min(DL_{n-m+1})], \quad (16)$$

где $DL_i (1 \leq i \leq n - m + 1)$.

i-й элемент в PL говорит нам о расстоянии от $T_{i,m}$ до его ближайшего левого соседа в T, но не говорит, где находится этот левый сосед. Эта информация хранится в сопутствующем векторе, называемом индексом левого МП.

Индекс левого МП IL представляет собой вектор целых чисел и представлен в формуле (17):

$$IL = [IL_1, IL_2, \dots, IL_{n-m+1}], \quad (17)$$

где $IL_i = j$, если $LNN(T_{i,m}) = T_{j,m}$.

Правый МП PR временного ряда T представляет собой вектор евклидова расстояния между каждой подпоследовательностью $T_{i,m}$ и ее правым ближайшим соседом во временном ряду. Формальное определение представлено в формуле (18):

$$PR = [min(DR_1), min(DR_2), \dots, min(DR_{n-m+1})], \quad (18)$$

где $DR_i (1 \leq i \leq n - m + 1)$.

Индекс правого МП IR представляет собой вектор целых чисел и представлен в формуле (19):

$$IR = [IR_1, IR_2, \dots, IR_{n-m+1}], \quad (19)$$

где $IR_i = j$, если $RNN(T_{i,m}) = T_{j,m}$.

В работе [28] представлено определение цепочки (chain) и алгоритмы ее поиска. Формальное определение цепочки представлено в формуле (20). Цепочка временного ряда T представляет собой упорядоченное множество подпоследовательностей:

$$TSC = \{T_{C1,m}, T_{C2,m}, \dots, T_{Ck,m}\}, \quad (20)$$

где $\forall\, i \in 1 \dots k \quad (C1 \leq C2 \leq \cdots \leq Ck)$, есть $RNN(T_{Ci,m}) = T_{C(i+1),m}$ и $LNN(T_{C(i+1),m}) = T_{Ci,m}$. k – длина цепочки.

Цепочки подразделяются на закрепленные и незакрепленные. Незакрепленная цепочка – это самая длинная цепочка во временном ряде, и она единственная, а закрепленных цепочек может быть много.

Множество всех цепочек (All-Chain Set) S – это множество всех закрепленных цепочек, каждая из которых не включена в какую-либо другую цепочку.

1.2. Обзор работ

Для интеллектуального анализа временных рядов есть множество различных готовых решений. Они предоставляют различные возможности, такие как интеллектуальный анализ, хранение, визуализация и их комбинации. Ниже пройдемся по каждому из пунктов.

Stumpy [23] – это мощная и масштабируемая библиотека Python, она эффективно вычисляет МП, который может использоваться для различных задач интеллектуального анализа данных временных рядов. Данная библиотека имеет множество различных методов для поиска примитивов. С помощью Stumpy можно найти мотивы, диссонансы, цепочки и многое другое.

Matplotlib [16] – это библиотека для создания графиков и визуализации данных на языке программирования Python, используемая для создания статических, анимированных и интерактивных визуализаций. Основная цель Matplotlib – предоставить пользователям разнообразные инстру-

менты для создания качественных графиков различных типов, начиная от простых линейных до сложных трехмерных и функциональность для графического представления данных, упрощая их анализ и понимание. В настоящее время поддерживается большим сообществом разработчиков.

Grafana [9] – это решение для аналитики и мониторинга с открытым исходным кодом, разработанное Grafana Labs, которое позволяет эффективно работать с данными, включая и временные ряды. Grafana предоставляет следующий набор функций.

1. Grafana предлагает широкий спектр возможностей визуализации, начиная от графиков и заканчивая гистограммами, позволяя пользователям глубже разбираться в своих данных.

2. Grafana упрощает управление оповещениями, позволяя пользователям визуально определять пороговые значения.

3. Grafana позволяет пользователям объединять данные из различных источников, предоставляя всеобъемлющий контекст для анализа. Встроенная поддержка десятков баз данных обеспечивает плавную интеграцию.

4. Пользователи могут создавать информативные отчеты и делиться ими.

InfluxDB [13] – это распределенная временная база данных, специализированная на хранении и обработке временных данных. Она широко используется в системах мониторинга, аналитике временных рядов и других приложениях, где требуется эффективное управление временными данными. InfluxDB доступна на всех популярных языках и фреймворках, что позволяет разработчикам приступить к работе за считанные минуты и располагает крупнейшим сообществом разработчиков облачных решений и решений с открытым исходным кодом для баз данных временных рядов. Позволяя сообществу вносить свой вклад в другие экосистемы и интегрироваться с ними, InfluxDB обладает большей устойчивостью, качеством,

возможностью использования и конфигурирования во всех вариантах использования.

ClickHouse [3] – высокопроизводительная, ориентированная на столбцы база данных с открытым исходным кодом, разработанная для обработки больших объемов данных и для оперативной аналитической обработки, которая использует все доступные системные ресурсы в полной мере для максимально быстрой обработки каждого аналитического запроса. Она доступна как в виде программного обеспечения с открытым исходным кодом, так и в виде облачного приложения.

TimescaleDB [24] – это расширение для базы данных PostgreSQL, специализированное на хранение временных данных. TimescaleDB объединяет мощность реляционной базы данных PostgreSQL с возможностями хранения и анализа временных рядов. Благодаря своей специализации на временных данных, TimescaleDB обеспечивает высокую производительность и масштабируемость при работе с данными, обладая при этом всеми преимуществами PostgreSQL.

Graphite [10] – это инструмент для сбора и визуализации метрик и временных рядов, который одинаково хорошо работает на дешевом оборудовании или облачной инфраструктуре. Он обладает мощными возможностями агрегации данных, построения графиков для мониторинга производительности системы.

Azure Time Series Insight (TSI) [2] – это сервис временных рядов в облаке от Microsoft Azure, предназначенный для хранения, анализа и визуализации временных данных. Azure TSI обеспечивает возможности работы с большими объемами данных, мониторинга и аналитики. С помощью Azure TSI пользователи могут проводить глубокий анализ данных временных рядов, исследовать тренды, выявлять аномалии и принимать оперативные решения на основе этих данных.

Real Time Intelligence Desktop (RTID) [20] – это настольное приложение, предназначенное для сбора, хранения, визуализации и анализа данных в режиме реального времени. Некоторые из ключевых преимуществ RTID.

1. Статистические данные собираются в режиме реального времени, что позволяет пользователям отслеживать показатели в режиме онлайн, анализировать реакцию на различные виды воздействия, просматривать историю и проводить сравнительный анализ. Данные хранятся локально, что обеспечивает защиту от рисков недоступности данных. Также программное обеспечение можно легко настроить на использование внешней базы данных для хранения.

2. Гибкая и быстрая настройка профилей сбора данных. Эта функция особенно полезна, когда скорость имеет первостепенное значение и пользователям необходимо быстро собрать специальные статистические данные для более детальной оценки характеристик системы.

3. RTID имеет удобный интерфейс, который облегчает работу пользователей с метриками даже без знания SQL.

Выводы по первому разделу

В данном разделе представлены нотации и формальные определения, а также проведен обзор работ, связанных с интеллектуальным анализом временных рядов.

2. ПРОЕКТИРОВАНИЕ

2.1. Определение требований

В ходе проектирования системы были определены следующие функциональные и нефункциональные требования.

Функциональные требования

Функциональные требования определяют функциональность программного обеспечения, то есть описывают, какое поведение должна предоставлять система:

1) регистрация и аутентификация:

– пользователи могут регистрироваться в системе, указывая уникальное имя пользователя, адрес электронной почты и пароль;

– пользователи могут авторизоваться в системе, используя свои учетные данные;

2) управление проектами:

– создание нового проекта с возможностью указания названия, описания и других дополнительных параметров;

– удаление проекта с подтверждением действия;

– редактирование проекта с изменением параметров;

– просмотр списка проектов пользователя с возможностью перехода в каждый из них;

3) управление данными:

– загрузка временного ряда в проект из файла формата .txt;

– отображение временных рядов на графике;

– поиск примитивов временного ряда;

– отображение найденных примитивов на графиках.

Нефункциональные требования

К нефункциональным требованиям системы относятся свойства, которыми она должна обладать.

1. Система должна обеспечивать хранение учетных данных пользователей и загруженных временных рядов, найденных примитивов.
2. Веб-приложение должно иметь интуитивно понятный пользовательский интерфейс.
3. Веб-приложение должно быть отзывчивым и обеспечивать быструю загрузку временных рядов и найденных примитивов.
4. Основная программная реализация осуществляется с использованием Python.

2.2. Варианты использования системы

Для проектирования системы был использован язык графического описания для объектного моделирования UML. Была построена модель взаимодействия пользователя с веб-приложением в виде диаграммы вариантов использования. В ходе анализа разрабатываемого приложения были выявлены основные варианты использования (рисунок 1).

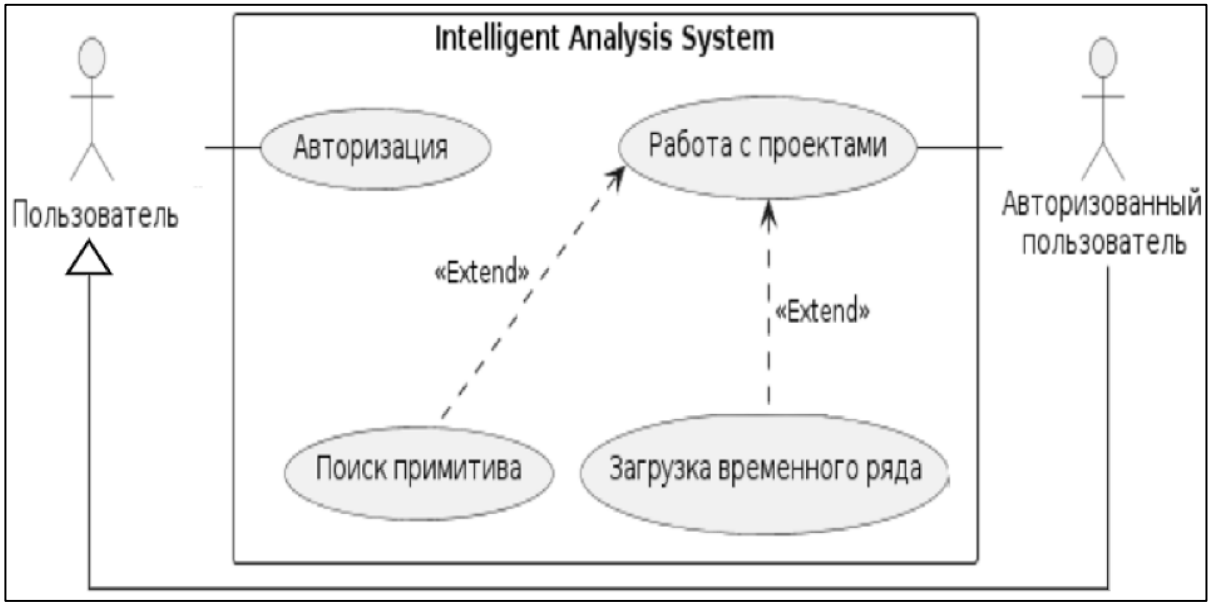

Рисунок 1 – Диаграмма вариантов использования

Для данной диаграммы определены актеры – пользователь, который может авторизоваться и авторизованный пользователь, который взаимодействует с веб-приложением. Пользователю доступны следующие варианты использования.

1. Вариант использования «Авторизация». Пользователь может зарегистрироваться в системе. Если он ранее был зарегистрирован, то можно войти в свою учетную запись.

2. Вариант использования «Работа с проектами». Пользователь может создать, редактировать, удалить и открыть проект. При создании проекта задаются его параметры, загружается временной ряд и происходит поиск примитивов. При редактировании проекта можно изменить его параметры проекта, загрузить дополнительные координаты временного ряда, редактировать параметры временного ряда. При открытии проекта можно запустить процесс поиска примитивов, влиять на отображение координат временного ряда и примитивов на графике.

3. Вариант использования «Загрузка временного ряда». Пользователь может запустить процесс загрузки, выполняющий считывание .txt файла, обработку его содержимого, с последующей загрузкой в базу данных.

4. Вариант использования «Поиск примитива». Пользователь может запустить процесс поиска top-k примитивов, выполняющий задание параметров, с последующим их поиском и загрузкой в базу данных.

2.3. Архитектура веб-приложения

В данном разделе рассматривается спроектированная архитектура веб-приложения в виде диаграммы компонентов, которая показывает разбиение системы на структурные компоненты.

Задача создания веб-приложения разбивается на две подзадачи: проектирование клиентской части и проектирование серверной части.

Серверная часть отвечает за обработку запросов, приходящих с клиентских устройств. Обработка запросов в большинстве случаев включает в себя взаимодействие сервера с базой данных, хранящей информацию, доступную для просмотра на сайте.

Клиентская часть отвечает за вывод информации, полученной от сервера, в виде веб-страниц в браузере пользователя, с интерфейсом которых он имеет возможность взаимодействовать.

Спроектированная архитектура системы представлена на рисунке 2 в виде диаграммы компонентов.

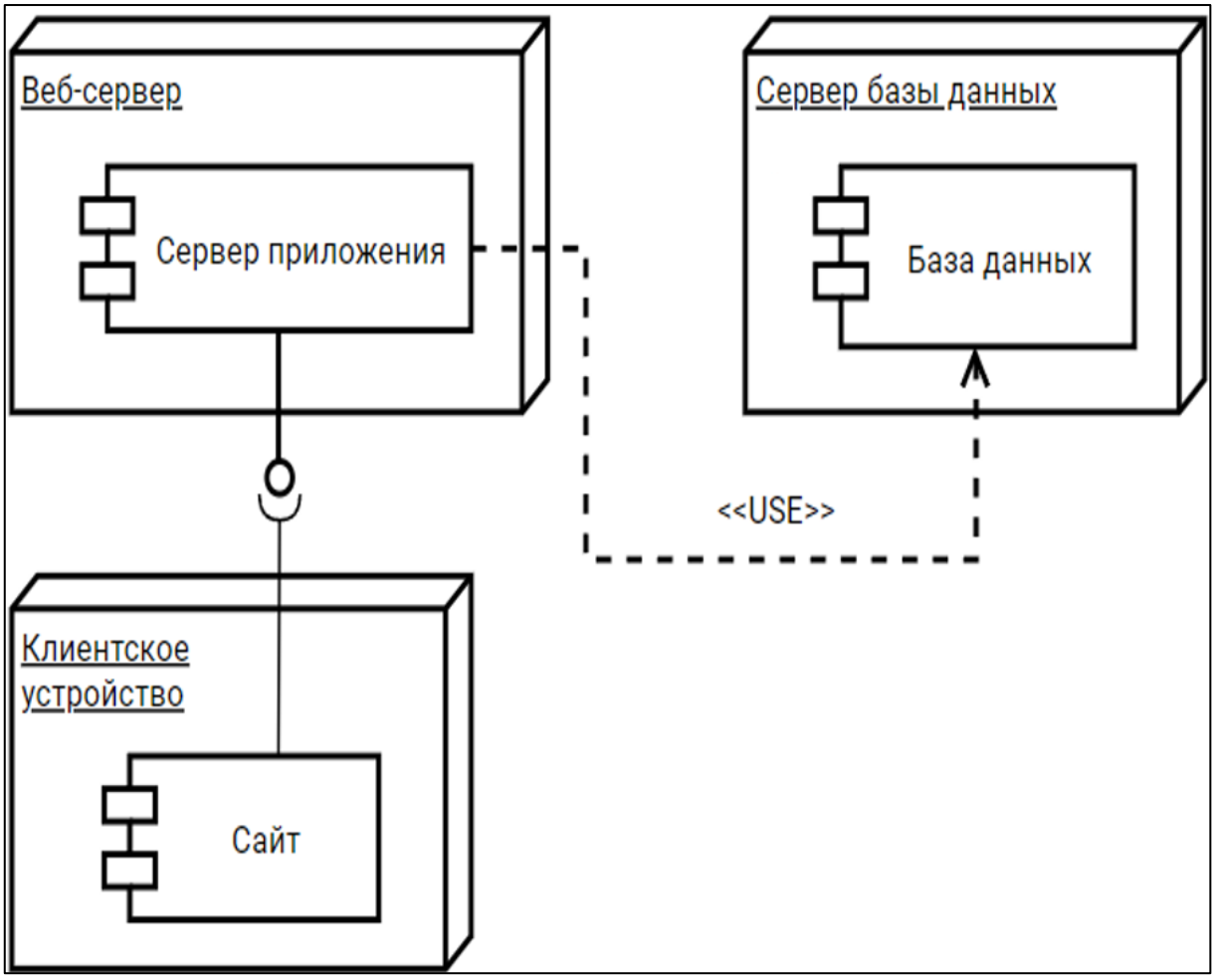

Рисунок 2 – Архитектура системы

2.4. Проектирование базы данных

База данных для веб-приложения должна хранить данные пользователя, проекта, временного ряда, МП, мотива, диссонанса, сниппета, цепоч-

ки, найденных примитивов. Все эти данные можно представить в виде отдельных сущностей. На основе этого была спроектирована entity relationship diagram (сокращенно ER-диаграмма или диаграмма сущность-связь), которая представлена на рисунке 3.

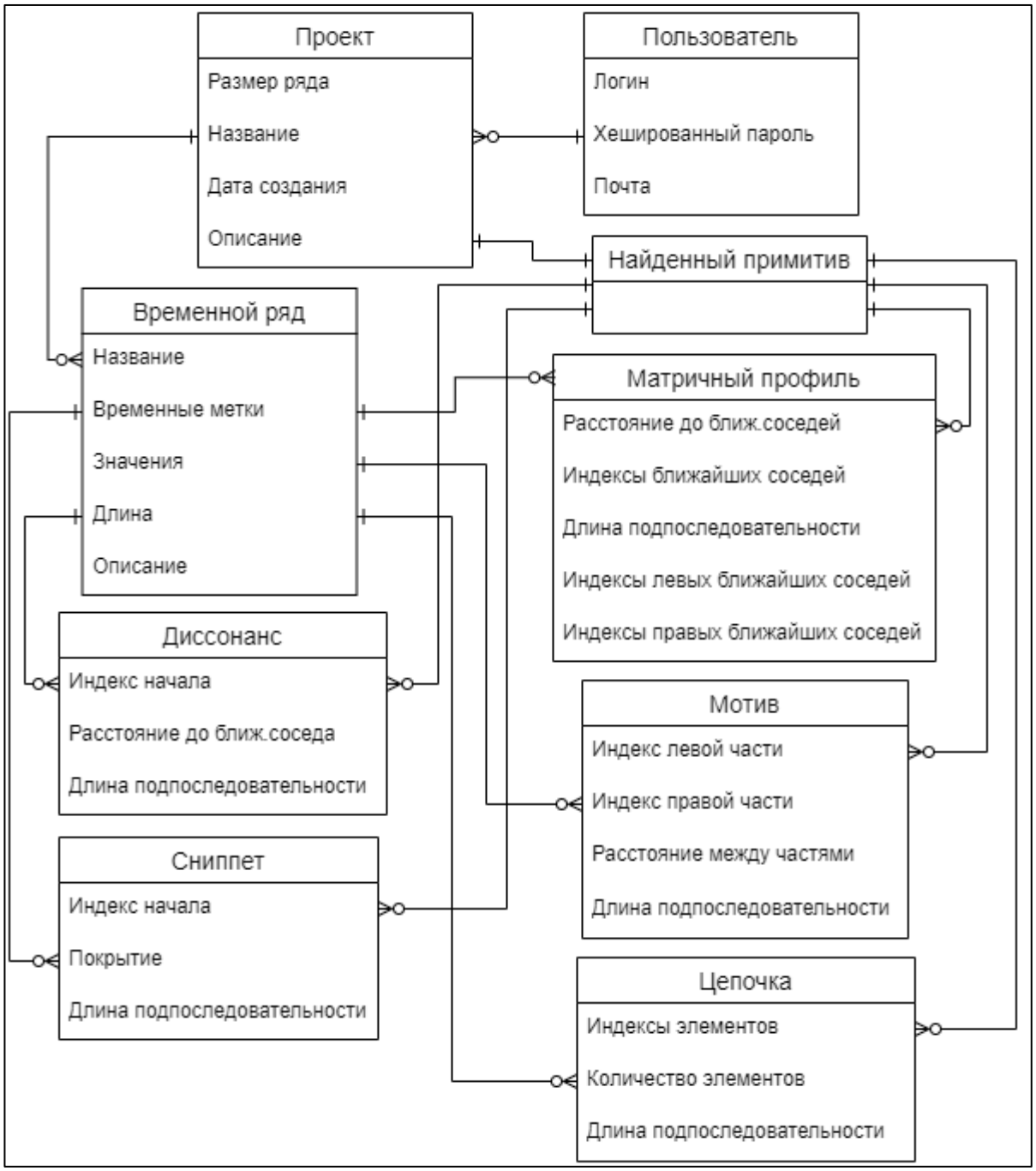

Рисунок 3 – Диаграмма сущность-связь

В соответствии с ER-диаграммой была спроектирована схема базы данных, которая представлена на рисунке 4.

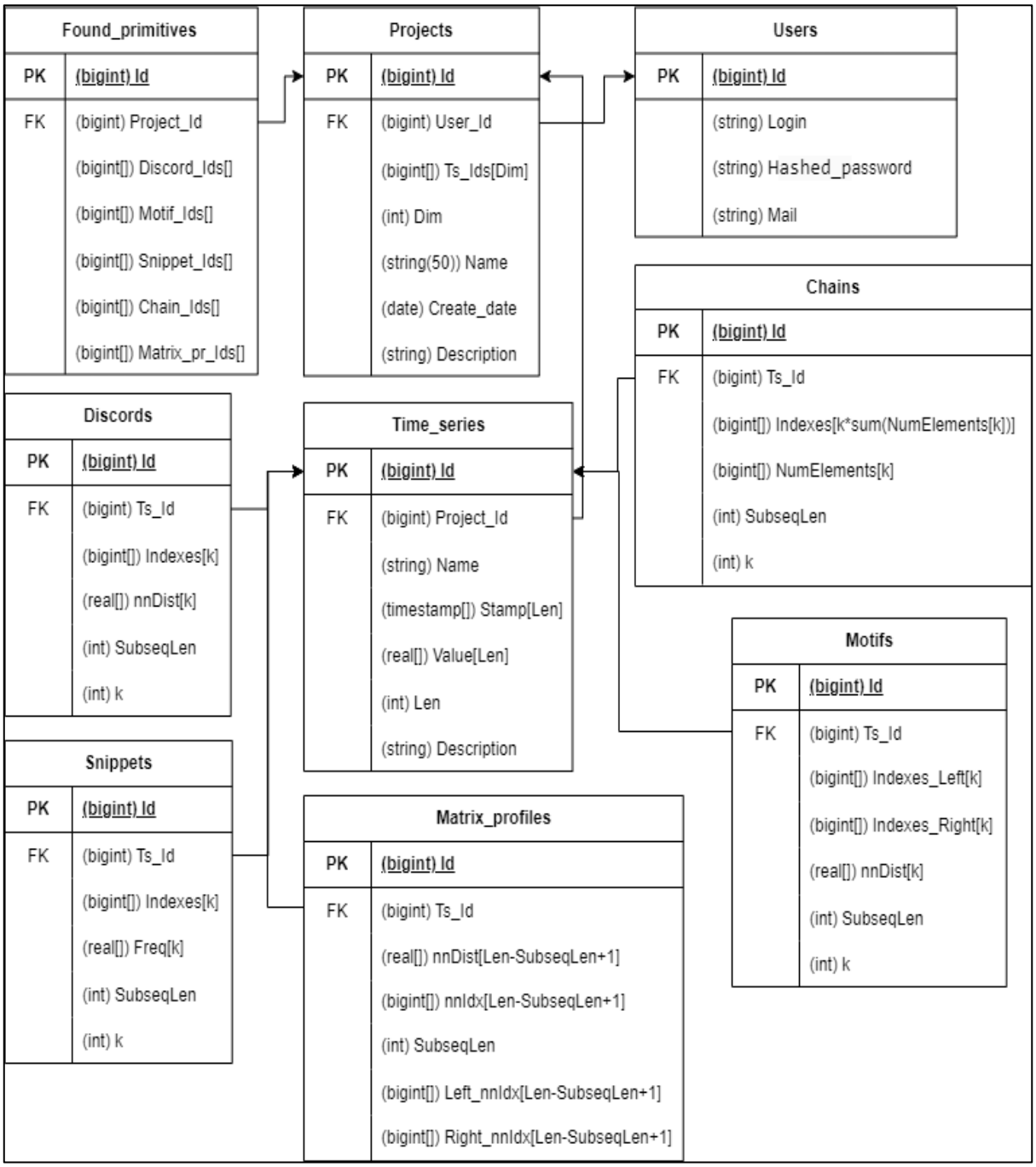

Рисунок 4 – Схема базы данных

База данных включает в себя 9 таблиц: «Users», «Projects», «Time_series», «Matrix_profiles», «Motifs», «Discords», «Snippets», «Chains», «Found_primitives».

Ниже дано краткое описание каждой из таблиц базы данных.

Таблица «Users» предназначена для хранения информации о зарегистрированных пользователях. Она содержит такие столбцы: уникальный идентификатор, логин, хешированный пароль и почту.

Таблица «Projects» предназначена для хранения информации о созданных проектах пользователей. Она содержит такие столбцы: уникальный идентификатор, размер временного ряда (если 1, то одномерный ряд, если больше, то многомерный), название, дату создания, описание, внешний ключ на пользователя, которому принадлежит проект, массив внешних ключей на координаты временного ряда.

Таблица «Time_series» предназначена для хранения информации о временных рядах. Если ряд одномерный, то одна строка хранит информацию о временном ряде, но если ряд многомерный, то одна строка хранит информацию о координатах временного ряда. Она содержит такие столбцы: уникальный идентификатор, название, массив временных меток, массив значений, длину и описание, а также внешний ключ на проект.

Таблица «Matrix_profiles» предназначена для хранения информации о найденных матричных профилях временных рядов. Она содержит такие столбцы: уникальный идентификатор, массив расстояний до ближайших соседей, массив индексов ближайших соседей, массив индексов левых ближайших соседей, массив индексов правых ближайших соседей, длину подпоследовательности, а также внешний ключ на координаты временного ряда.

Таблица «Motifs» предназначена для хранения информации о найденных мотивах. Она содержит такие столбцы: уникальный идентификатор, массивы индексов левой и правой части мотивов, массив расстояний между частями, длину подпоследовательности, количество, а также внешний ключ на координаты временного ряда. Каждая строка таблицы хранит информацию о top-k мотивах временного ряда с заданной длиной подпоследовательности.

Таблица «Discords» предназначена для хранения информации о найденных диссонансах. Она содержит такие столбцы: уникальный идентификатор, массив индексов начала, массив расстояний до ближайших соседей, длину подпоследовательности, количество, а также внешний ключ на координаты временного ряда. Каждая строка таблицы хранит информацию о top-k диссонансах временного ряда с заданной длиной подпоследовательности.

Таблица «Snippets» предназначена для хранения информации о найденных сниппетах. Она содержит такие столбцы: уникальный идентификатор, массив индексов начала, массив покрытий, длину подпоследовательности, количество, а также внешний ключ на координаты временного ряда. Каждая строка таблицы хранит информацию о top-k сниппетах временного ряда с заданной длиной подпоследовательности.

Таблица «Chains» предназначена для хранения информации о найденных цепочках. Она содержит такие столбцы: уникальный идентификатор, массив индексов начала, массив количества элементов, длину подпоследовательности, количество, а также внешний ключ на координаты временного ряда. Каждая строка таблицы хранит информацию о top-k цепочках временного ряда с заданной длиной подпоследовательности.

Таблица «Found_primitives» предназначена для хранения информации о найденных примитивах. Она содержит такие столбцы: уникальный идентификатор, внешний ключ на проект, массив внешних ключей найденных диссонансов, массив внешних ключей найденных мотивов, массив внешних ключей найденных сниппетов, массив внешних ключей найденных цепочек, массив внешних ключей найденных матричных профилей. В приложение А вынесена детальная структура описанных выше таблиц.

2.5. Проектирование пользовательского интерфейса

В данном разделе будут представлены основные спроектированные макеты пользовательского интерфейса веб-приложения.

Данные макеты являются примерным представлением итогового продукта и содержат в себе основные необходимые функции. Для разрабатываемого веб-приложения было решено создать простой и интуитивно понятный интерфейс для удобства пользователя.

После варианта использования «Авторизация» пользователь попадает на основную веб-страницу, которая изображена на рисунке 5.

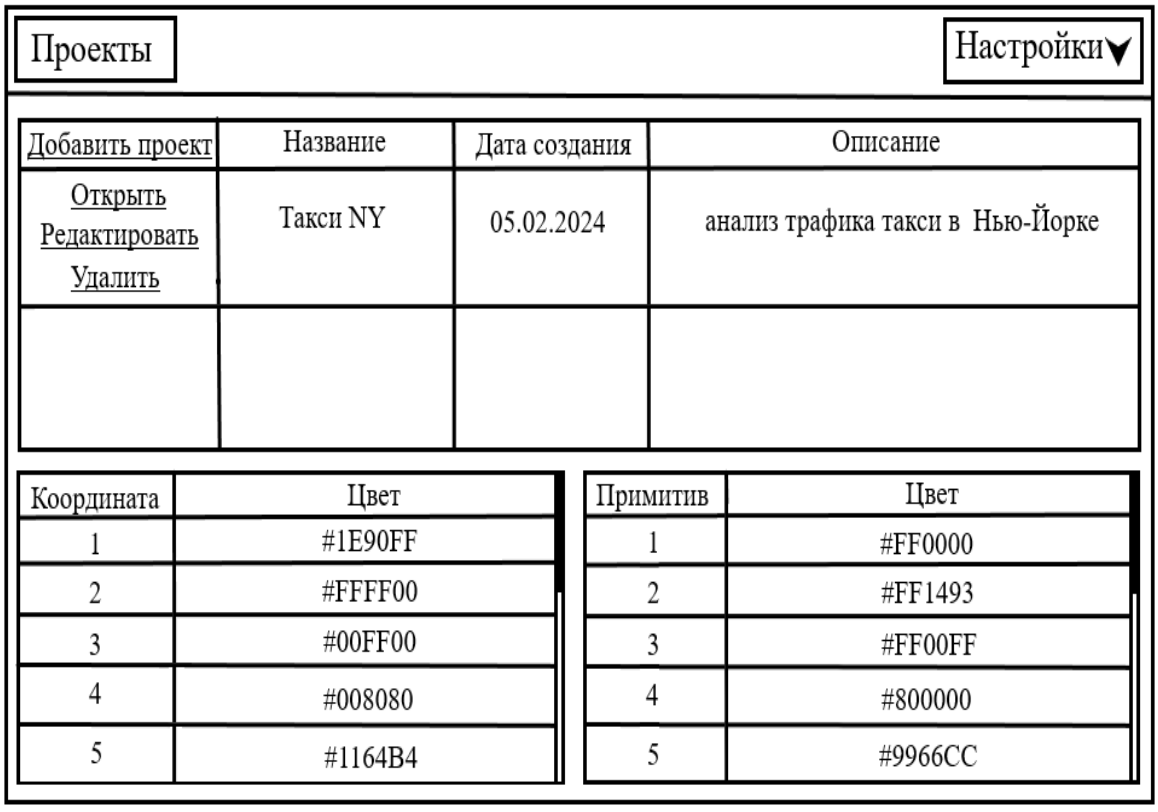

Рисунок 5 – Макет основной формы веб-приложения

Данная веб-страница реализует вариант использования «Работа с проектами». На ней расположены: ссылка «Проекты», которая будет возвращать пользователя на данную веб-страницу, список «Настройки» через который можно выйти из учетной записи, две таблицы, в которых можно настроить цвета первым 10 наборам координат и примитивов, а также есть

таблица проектов, в которой показывается основная информация, связанная с проектами пользователя.

На основной веб-странице можно открыть форму создания и редактирования проекта, при нажатии по соответствующей ссылке, также можно удалить определенный проект или перейти на веб-страницу выбранного проекта.

На рисунке 6 представлен макет формы создания проекта.

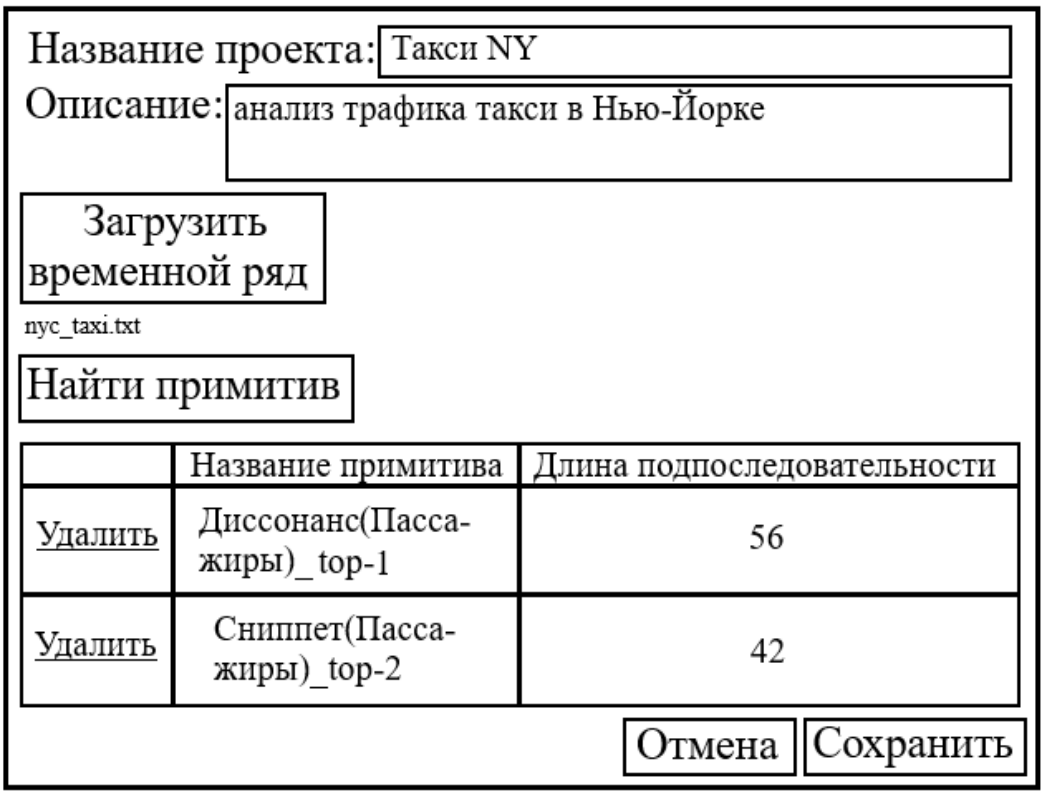

Рисунок 6 – Макет формы создания проекта

На данной форме задаются название и описание проекта, при нажатии на «Загрузить временной ряд» запускается вариант использования «Загрузка временного ряда», при нажатии на «Найти примитив» запускается вариант использования «Поиск примитива», в таблице отображаются примитивы, которые начнут искаться после нажатия на кнопку «Сохранить».

На рисунке 7 представлен макет формы поиска примитива.

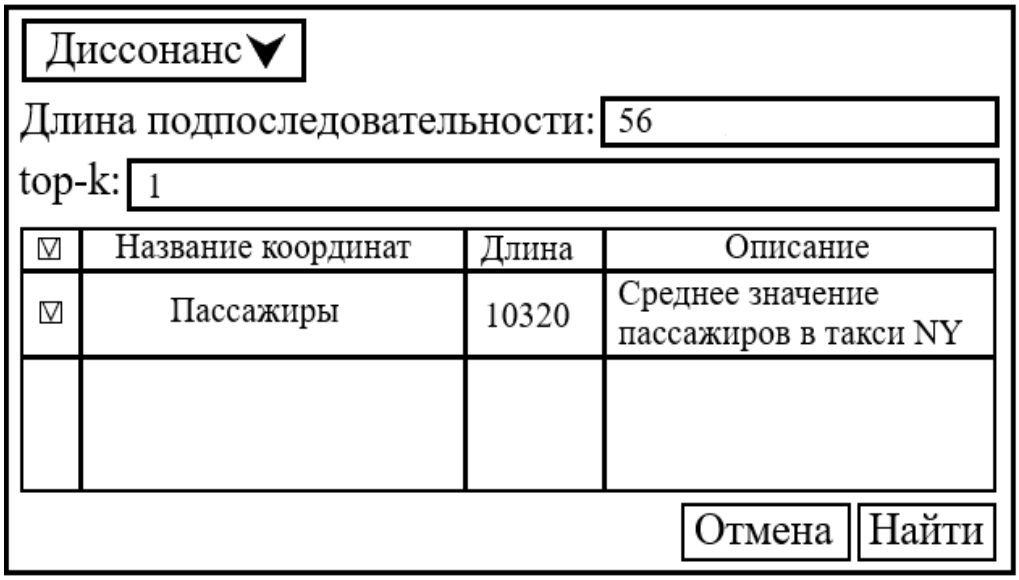

Рисунок 7 – Макет формы поиска примитива

В данной форме можно выбрать из списка примитив, который хотим найти, задать длину подпоследовательности и сколько примитивов хотим найти, также есть таблица с координатами временного ряда, в которой нужно выбрать, где будет происходить поиск примитивов.

На рисунке 8 представлен макет формы редактирования проекта.

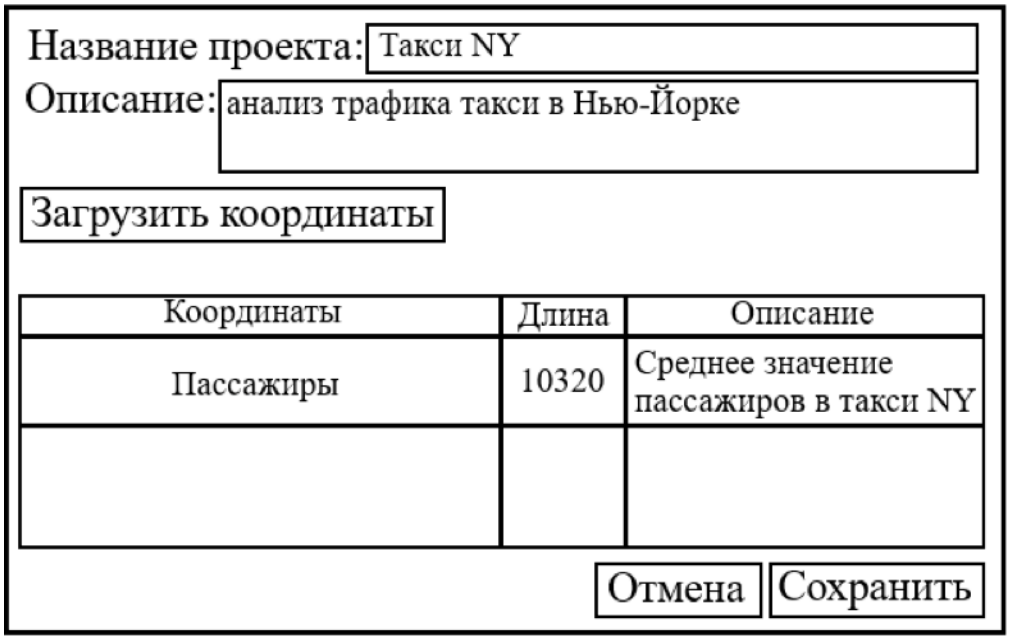

Рисунок 8 – Макет формы редактирования проекта

При редактировании проекта, можно изменить название и описание проекта, посмотреть таблицу с координатами временного ряда и загрузить еще координат, чтобы увеличить размерность.

На рисунке 9 представлен макет формы выбранного проекта.

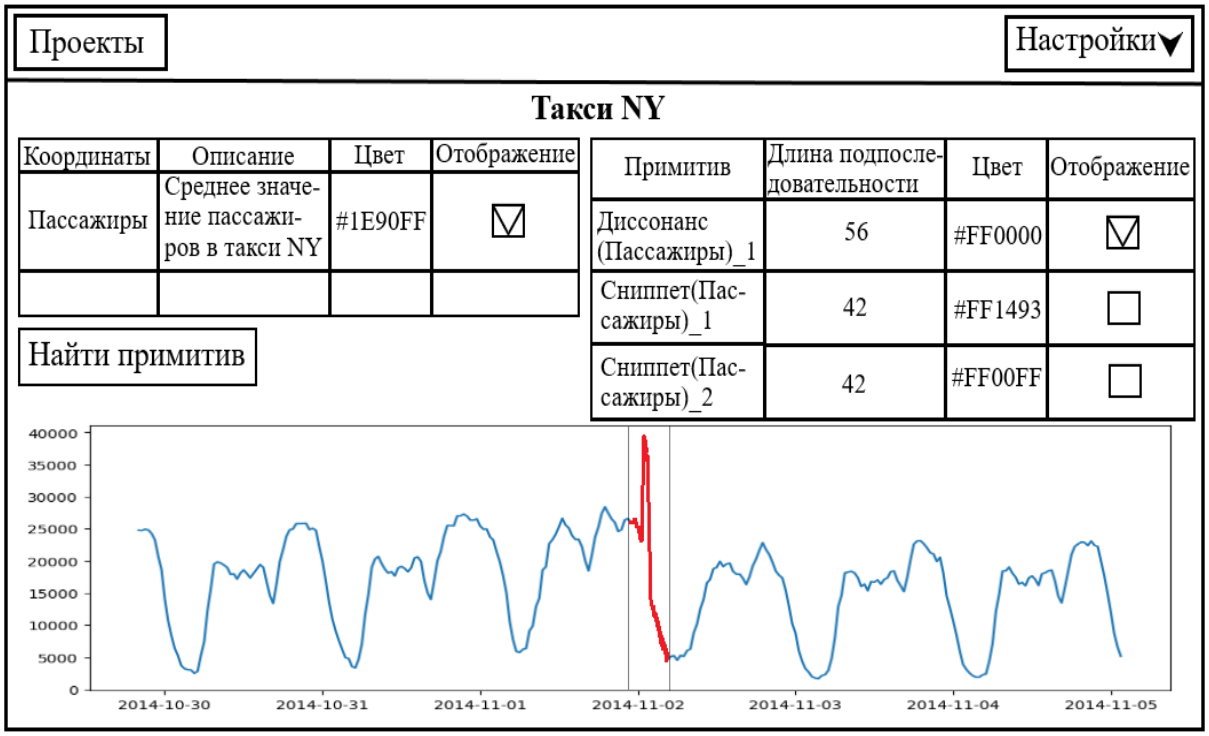

Рисунок 9 – Макет формы выбранного проекта

На данной форме у нас есть график, на котором изображены координаты и примитивы временного ряда, таблица с координатами временного ряда, где отображается описание и можно отобразить или скрыть их на графике, еще одна такая же таблица, только о примитивах, также при нажатии на «Найти примитив» запустится вариант использования «Поиск примитива».

Остальные макеты форм представлены в приложении Б.

Выводы по второму разделу

В данном разделе исходя из поставленных задач были разработаны функциональные и нефункциональные требования к оболочке, построена диаграмма вариантов использования, спроектирована база данных, спроектирован интерфейс системы, а также разработана архитектура системы.

3. РЕАЛИЗАЦИЯ

Программные средства реализации

Чтобы реализовать оболочку для автоматизации выполнения интеллектуального анализа временных рядов было решено использовать следующий стек технологий:

1) Python 3.12 – язык разработки серверной части;
2) FastApi [6]– фреймворк для разработки серверной части;
3) Stumpy – библиотека, с помощью которой произведен поиск примитивов;
4) PostgreSQL [18] – база данных;
5) PyCharm Community Edition 2023.3.5 – среда разработки серверной части;
6) JavaScript – язык разработки веб-интерфейса;
7) Vue [25] – фреймворк для разработки веб-интерфейса;
8) DevExtreme [4] – набор инструментов разработки веб-приложений, предоставляющий большое количество различных компонентов;
9) Visual Studio Code – среда разработки веб-интерфейса.

3.1. Реализация API

API – Application Programming Interface, что значит программный интерфейс приложения. API называется определенный набор протоколов, подпрограмм и инструментов для создания программным приложений. Основной особенностью API является его способность настраивать взаимодействие компонентов различных информационно-аналитических систем, т.е. он обеспечивает эффективный процесс коммуникаций между программами, которые используют функции и ресурсы друг друга.

Основным фреймворком для реализации оболочки является FastApi. FastAPI – это фреймворк для разработки RESTful API на Python.

Архитектурно выделено 3 слоя: слой репозитория, слой бизнес-логики и слой представлений. Одними из основных задач API является загрузка временных рядов и поиск примитивов. В листинге 1 показана обработка временного ряда.

Листинг 1 – Обработка временного ряда

```
df = pd.read_csv(file.file)
time_series = []
is_timestamp = df.columns[0].lower().strip() == 'timestamp'
if df.isnull().values.any() or df.shape[0] < 5 or is_timestamp and
df.shape[1] < 2:
      return [False, "Неверная структура файла"]
coord_names = df.columns
if is_timestamp:
      df.iloc[:, 0] = pd.to_datetime(df.iloc[:, 0])
      coord_names = df.columns[1:]
for title in coord_names:
      if (df[title]).dtypes == object:
            return [False, "Координаты должны быть типа float"]
      text = title
      if text.count('(') > 1 or text.count(')') > 1 or text.count('(') !=
      text.count(')'):
            return [False, "Неверная структура файла"]
      text = re.sub(r'\s+', ' ', text.strip()
      temp = re.findall(r'([^(]+)\(((\s*\S+\s*)+)+\)', text)
      if len(temp) == 0:
            temp = [text, None]
      else:
            temp = [temp[0][0].strip(), temp[0][1].strip()]
      time_series.append(TimeSeries(
            name=temp[0],
            stamp=df.iloc[:, 0] if is_timestamp else [],
            value=df[title],
            len=df.shape[0],
            description=temp[1]
      ))
```

Для поиска примитивов используется библиотека Stumpy. Поиск примитивов происходит в несколько этапов. Сначала получаем данные, а точнее список примитивов, который содержит: id координаты для которого нужно найти примитив, название примитива, длина подпоследовательности и число, обозначающее топ скольких примитивов нужно найти. После создается словарь, где ключ – это первая буква примитива, а значением является список примитивов из полученных данных, которые необходимо найти. Затем из базы данных получаем координаты временного ряда по id координат, для этих координат получаем матричные профили и найденные примитивы (например, если хотим найти top-k диссонансов, то из базы бе-

рем только найденные диссонансы). После берем значение по каждому ключу словаря и в цикле проверяем найден ли такой примитив ранее, если примитив был найден ранее, тогда берем следующий примитив. Иначе проверяем найден ли МП с такой длинной подпоследовательности, если он был найден, то используем его, иначе вычисляем МП. Затем выполняется поиск примитива, в конце все найденные примитивы записываются в базу данных.

В первую очередь был реализован слой репозитория. В качестве СУБД была выбрана PostgreSQL. В ней была создана структура таблиц, указанная в проектировании (раздел 2.4).

Запросы к базе данных осуществляются с помощью ORM SQL Alchemy [22]. SQLAlchemy – это библиотека, которая облегчает взаимодействие между программами на Python и базами данных. В большинстве случаев эта библиотека используется как средство объектно-реляционного отображения (ORM), которое преобразует классы Python в таблицы в реляционных базах данных и автоматически преобразует вызовы функций в инструкции SQL. Для миграций используется Alembic [1].

Для описаний схем и моделей использовалась библиотека Pydantic [19]. Она наиболее широко используемая библиотека проверки данных для Python. На рисунке 10 представлен один из примеров pydantic схемы.

```
class DgProject(BaseModel):
    id: int
    name: str
    create_date: datetime
    description: Optional[str]
    model_config = ConfigDict(from_attributes=True)
```

Рисунок 10 – Схема проекта

На рисунке 11 представлен пример описания модели таблицы.

```python
bigintPk = Annotated[int, mapped_column(BigInteger, primary_key=True)]
listBigint = Annotated[List[int], mapped_column(ARRAY(BigInteger))]



class Project(Base):
    __tablename__ = "projects"

    id: Mapped[bigintPk]
    user_id: Mapped[int] = mapped_column(ForeignKey( column: "users.id", ondelete="CASCADE"))
    ts_ids: Mapped[Optional[listBigint]]
    dim: Mapped[Optional[int]]
    name: Mapped[str] = mapped_column(String(30))
    create_date: Mapped[datetime] = mapped_column(server_default=text("TIMEZONE('utc', now())"))
    description: Mapped[Optional[str]]
```

Рисунок 11 – Модель таблицы проектов

Затем были реализованы маршруты API. Пример маршрута загрузки временного ряда представлен на рисунке 12.

```python
@app.post( path: "/file/upload-file", response_model=List[int])
async def upload_file(file: UploadFile) -> List[int]:
    res = await services.insert_time_series(file)
    if not res[0]:
        raise HTTPException(status_code=500, detail=res[1])
    return res
```

Рисунок 12 – Маршрут загрузки временного ряда

В результате был разработан REST API соответствующий спецификации openapi3. На рисунке 13 представлена спецификация маршрутов.

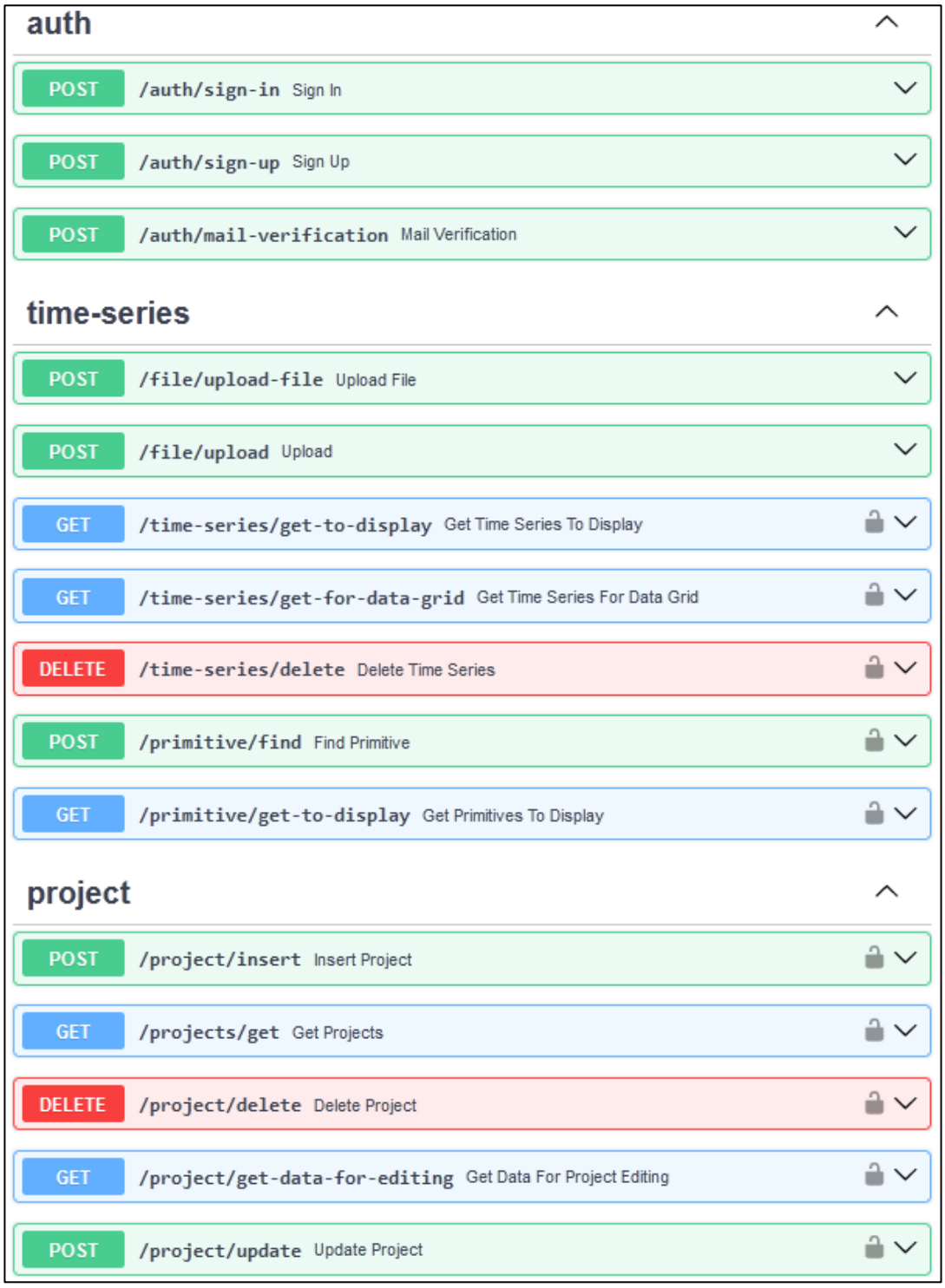

Рисунок 13 – Спецификация маршрутов

В качестве метода аутентификации используется аутентификация с помощью JWT-токенов. JWT-токены – это открытый стандарт для создания токенов доступа, основанный на формате JSON. Как правило, используется для передачи данных для аутентификации в клиент-серверных при-

ложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности.

3.2. Реализация пользовательского интерфейса

Для реализации оболочки был выбран фреймворк Vue.js. В качестве пакетного менеджера использовался npm. В качестве среды разработки был выбран Visual Studio Code. Основой для реализации веб интерфейса служит библиотека DevExtream – библиотека UI виджетов от компании DevExpress.

Все графики, формы и валидация форм были реализованы с помощью библиотеки DevExtream. Основная страница представлена на рисунке 14.

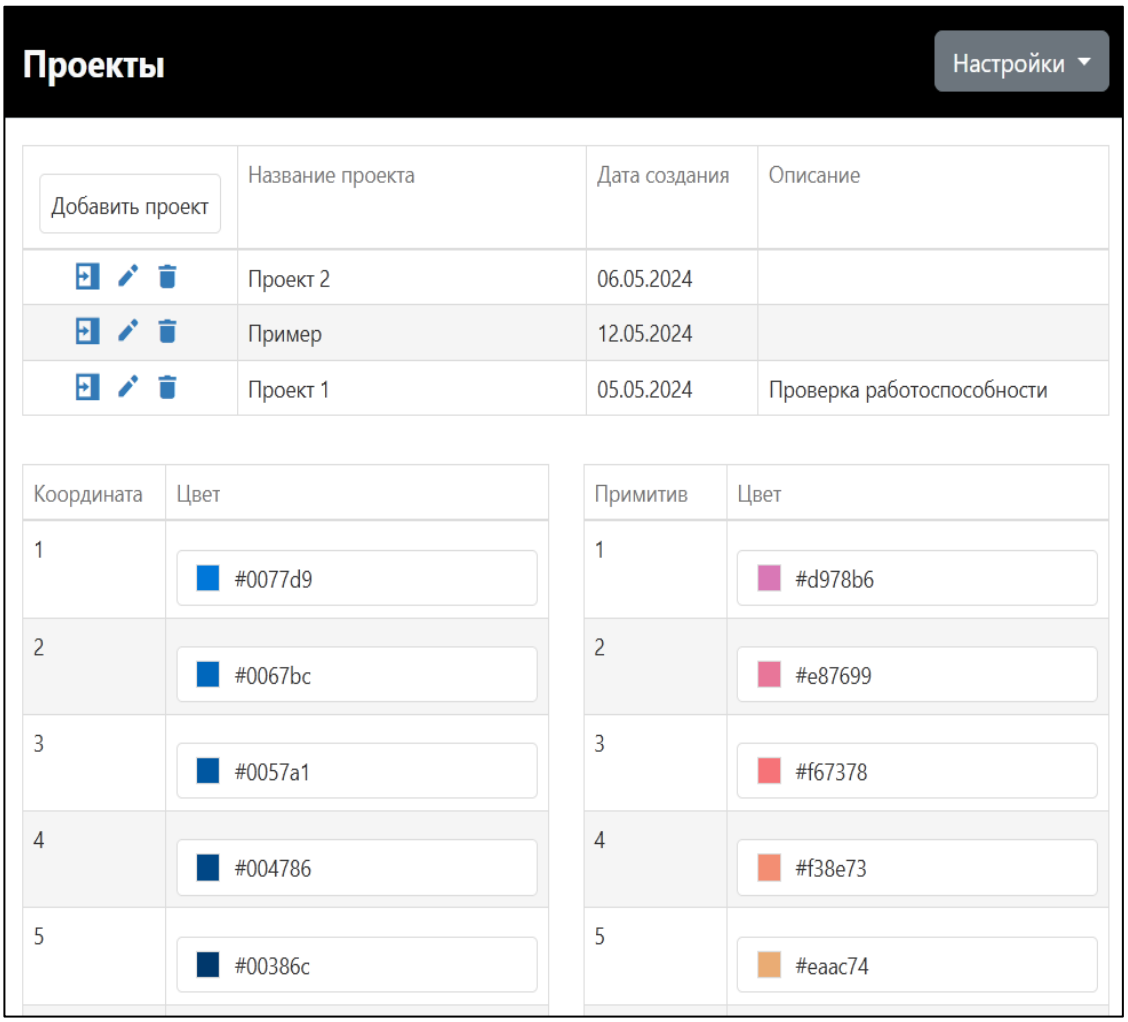

Рисунок 14 – Основная страница

На данной странице можно задать цвет первых 10 координат и примитивов. Также можно делать различные манипуляции с проектом.

На рисунке 15 представлен процесс создания проекта, а точнее когда был выбраны примитивы для поиска.

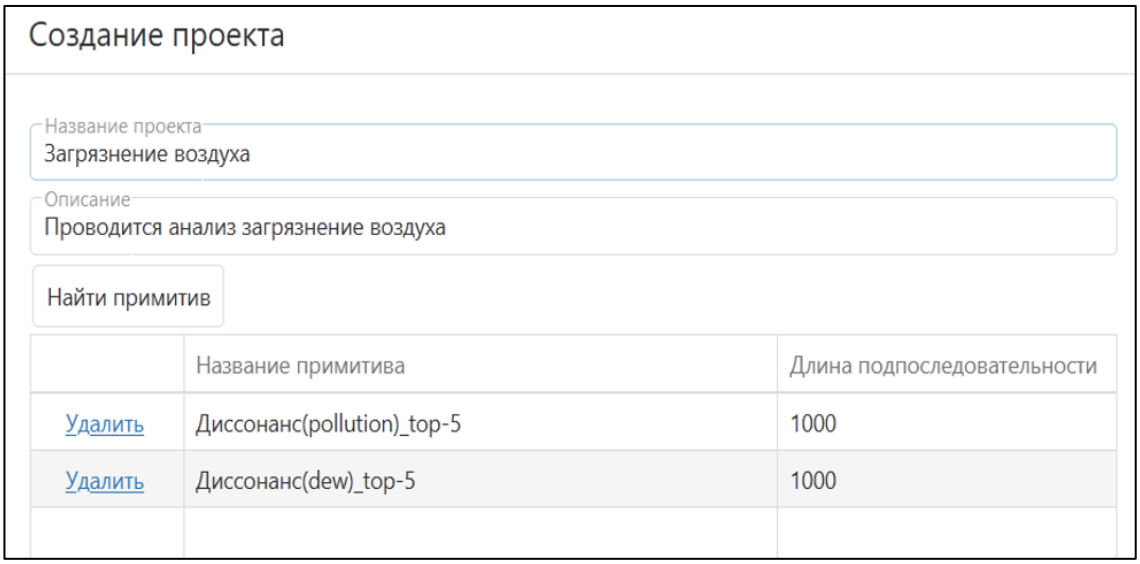

Рисунок 15 – Форма создания проекта

На рисунке 16 представлено то, как будет выглядеть выбор примитива для поиска.

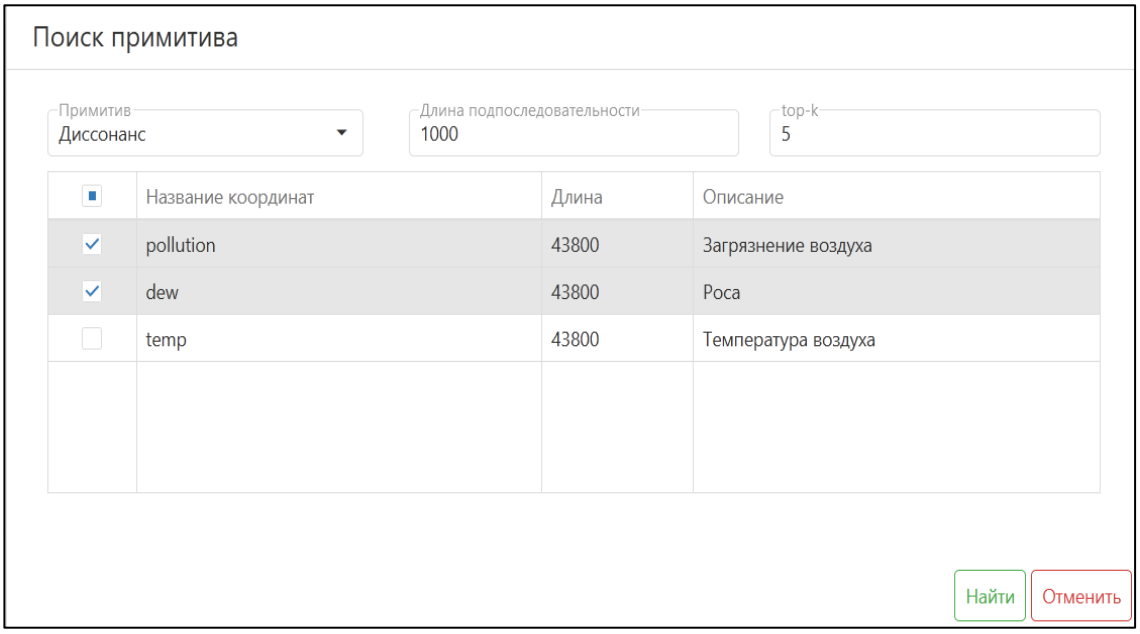

Рисунок 16 – Форма поиска примитива

После того как проект создан, в него загружен временной ряд и найдены примитивы, пользователь может открыть проект и увидеть графики. На рисунке 17 представлена страница выбранного проекта.

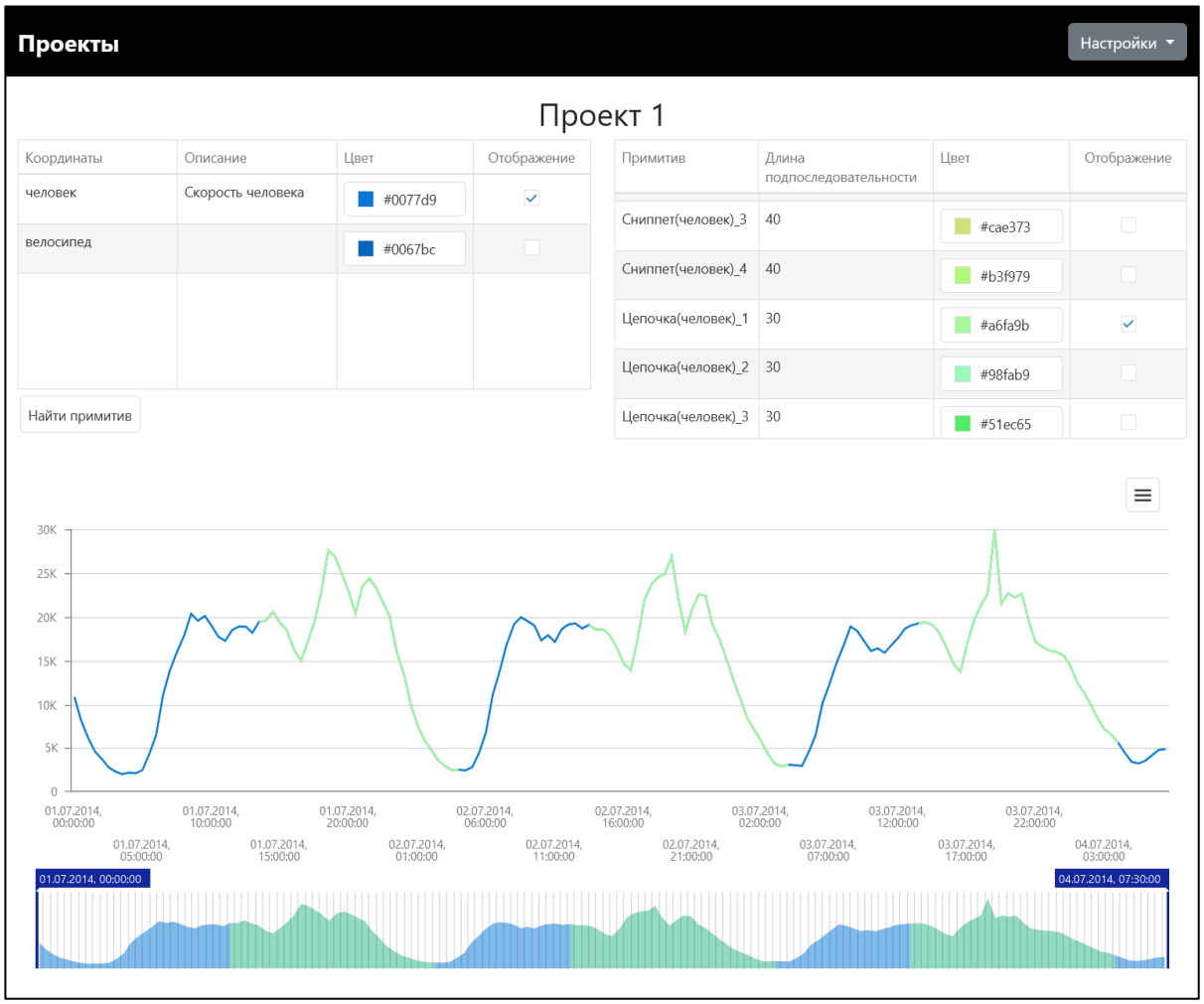

Рисунок 17 – Страница выбранного проекта

На данной странице можно изменять цвет, отображать или скрывать координаты временного ряда и примитивов. Так же доступно масштабирование графика и изменения диапазона меток. При нажатии на кнопку в правом углу графика, можно выбрать сохранение графика в нескольких форматах или распечатать.

Выводы по третьему разделу

В данной главе были представлены используемые технологии и результаты реализации. На основании функциональных требований были реализованы все компоненты.

4. ТЕСТИРОВАНИЕ И ЭКСПЕРИМЕНТЫ

4.1. Функциональное тестирование

В ходе тестирования проверялось соответствие веб-приложения функциональным требованиям. В таблице 1 приведены результаты функционального тестирования. Как видно по результатам тестирования из протокола, все функциональные тесты успешно пройдены.

Таблица 1 – Протокол функционального тестирования

№	Функциональное требование	Ожидаемый результат	Тест пройден?
1	Пользователи могут регистрироваться в системе	После регистрации пользователь попадает на основную страницу	Да
2	Пользователи могут авторизоваться в системе	После авторизации пользователь попадает на основную страницу	Да
3	Пользователь может создавать проект	При создании проекта открывается форма, в которой задаются параметры проекта, после сохранения, таблица проектов обновляется	Да
4	Пользователь может редактировать проект	Открывается форма редактирования, в котором моно изменить параметры проекта, после сохранения, пользователя таблица проектов обновляется	Да
5	Пользователь может удалить проект	Появляется всплывающее окно подтверждения удаления, при положительном ответе, удаляет проект и обновляет таблицу с проектами	Да
6	Пользователь может открыть проект	При выборе проекта, пользователь переходит на страницу выбранного проекта, с подзагрузкой соответствующих данных проекта	Да
7	Пользователь может искать примитивы временного ряда	При создании и на странице выбранного проекта можно задать параметры примитива и запустить его поиск.	Да
8	Пользователь может загружать временной ряд из файла формата .txt	При создании и редактировании проекта загружаются координаты временного ряда	Да
9	Пользователь может отображать временной ряд и примитивы	На странице выбранного проекта отображается график координат временного ряда и найденных примитивов	Да

4.2. Исследование функции поиска примитивов

Здесь представлены результаты исследования функции поиска примитивов, в виде зависимости времени выполнения от длины подпоследовательности.

Исследование проводилось на временной ряде, содержащем информацию о среднем числе пассажиров Нью-Йоркского такси за осень 2014 года. Временной ряд содержит 10320 значений.

На рисунке 18 представлена столбчатая диаграмма, на которой показана зависимость времени выполнения функции поиска примитива от длины подпоследовательности и примитива, когда МП не был вычислен.

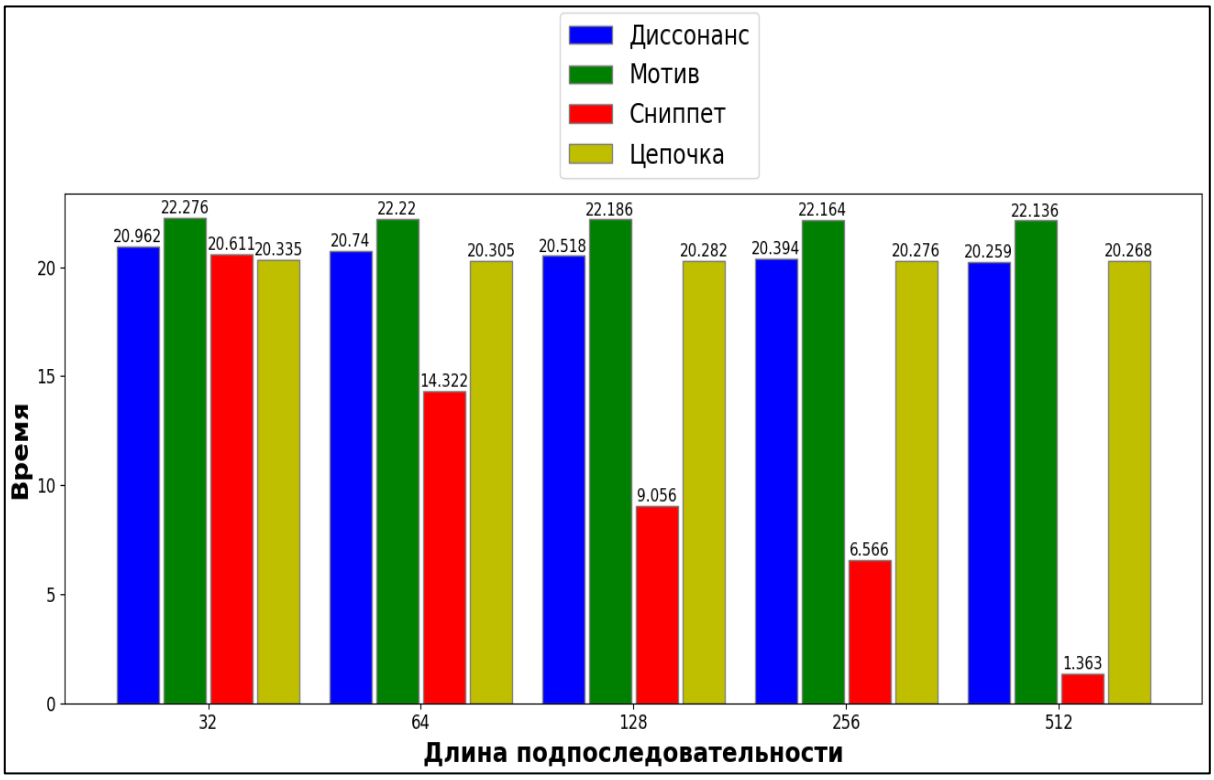

Рисунок 18 –Диаграмма при отсутствии вычисленного МП

На рисунке 19 представлена столбчатая диаграмма, на которой показана зависимость времени выполнения функции поиска примитива от длины подпоследовательности и примитива, когда МП был уже вычислен.

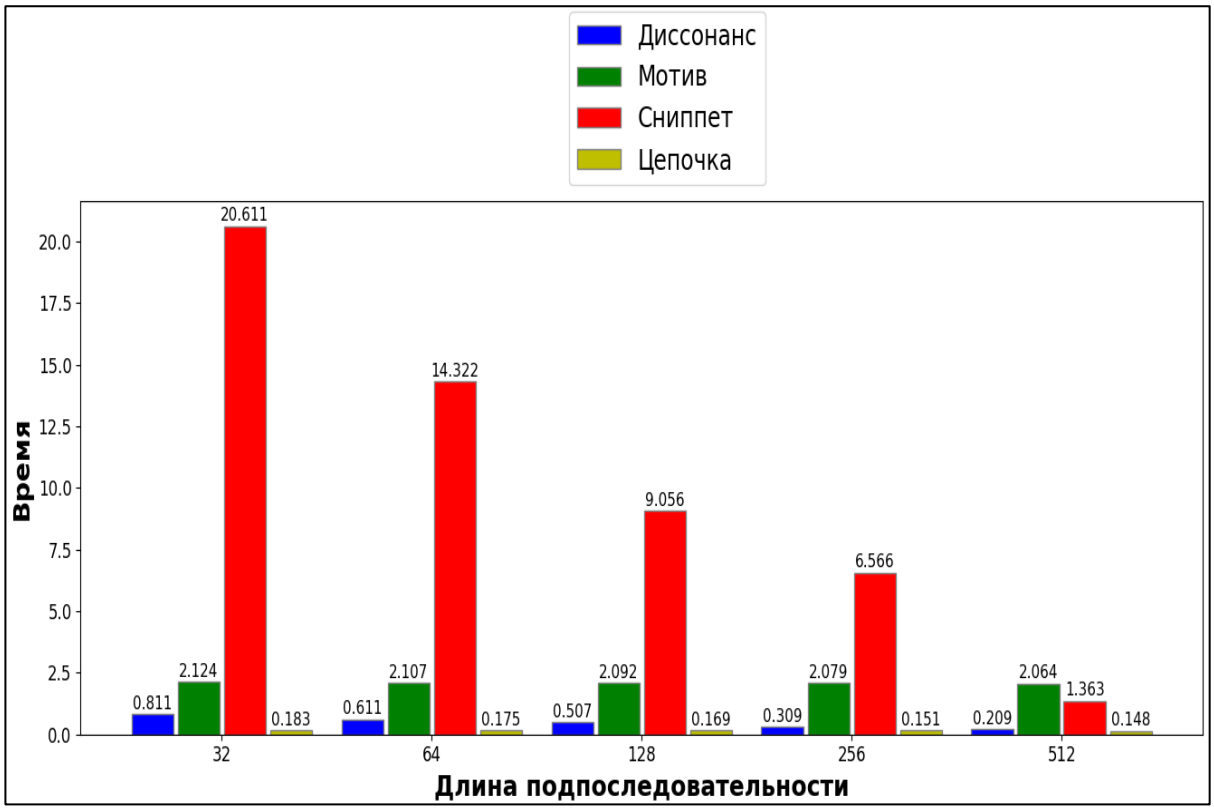

Рисунок 19 – Диаграмма с предвычисленным МП

Исходя из результатов, видно, что предвычисленный МП, дает выигрыш во времени поиска примитивов. Из всего этого выбивается только сниппеты, т.к. там алгоритм поиска не предусматривает МП.

Выводы по четвертому разделу

В данной главе приведены результаты функционального тестирования и исследования скорости выполнения функции поиска примитивов. На основании результатов можно сделать вывод о корректности работы системы. Система полностью удовлетворяет функциональным и нефункциональным требованиям, определенным на этапе проектирования, а также обеспечивает корректное представление результатов своей работы.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана оболочка для автоматизации выполнения интеллектуального анализа временных рядов.

При этом решены следующие задачи:

1) выполнен анализ предметной области и разработана спецификация требований к оболочке;

2) выполнено проектирование оболочки, включая веб-интерфейс пользователя, структуры базы данных и модульной структуры системы;

3) выполнены реализация, тестирование и отладка оболочки;

4) выполнены эксперименты по исследованию эффективности работы оболочки.

В будущем планируется продолжение разработки и улучшение оболочки, в частности добавление новых примитивов, более гибкую настройку поиска примитивов и улучшение веб-интерфейса пользователя.

Исходный код оболочки для автоматизации выполнения интеллектуального анализа временных рядов доступен по URL-адресу: https://github.com/00DarkNight00/Intelligent-Analysis-System.

ЛИТЕРАТУРА

1. Alembic. [Электронный ресурс] URL: https://pypi.org/project/alembic/ (дата обращения: 15.02.2024 г.).

2. Azure Time Series Insights. [Электронный ресурс] URL: https://www.sqlshack.com/introduction-to-azure-time-series-insights/ (дата обращения: 16.01.2024 г.).

3. ClickHouse. [Электронный ресурс] URL: https://db-engines.com/en/system/ClickHouse (дата обращения: 12.01.2024 г.).

4. DevExtreme. [Электронный ресурс] URL: https://js.devexpress.com/Vue/ (дата обращения: 22.02.2024 г.).

5. Esling P., Agon C. Time-series data mining. // ACM Comput. Surv., 2012. – Vol. 45, no. 1. – P. 12:1–12:34.

6. FastApi. [Электронный ресурс] URL: https://fastapi.tiangolo.com/ (дата обращения: 14.02.2024 г.).

7. Galton, F. Regression Towards Mediocrity in Hereditary Stature // Journal of the Anthropological Institute of Great Britain and Ireland, 1886. – 246–263 pp.

8. Gharghabi S., Imani S., Bagnall A.J., Darvishzadeh A., Keogh E.J. An ultra-fast time series distance measure to allow data mining in more complex real-world deployments. // Data Min. Knowl. Discov., 2020. – Vol. 34, no. 4. – 1104–1135 pp.

9. Grafana. [Электронный ресурс] URL: https://grafana.com/grafana/ (дата обращения: 12.01.2024 г.).

10. Graphite. [Электронный ресурс] URL: https://graphiteapp.org/ (дата обращения: 16.01.2024 г.).

11. Imani S., Madrid F., Ding W., Crouter S.E., Keogh E.J. Introducing time series snippets: a new primitive for summarizing long time series. // Data Min. Knowl. Discov, 2020. – 1713–1743 pp.

12. Imani S., Madrid F., Ding W., Crouter S.E., Keogh E.J., Wu X., Ong Y., Aggarwal C.C., Chen H. Matrix Profile XIII: Time Series Snippets: A New

Primitive for Time Series Data Mining. // 2018 IEEE International Conference on Big Knowledge, ICBK 2018, Singapore, November 17–18, 2018, IEEE Computer Society, 2018. – 382–389 pp.

13. Influx data. [Электронный ресурс] URL: https://www.influxdata.com/ (дата обращения: 12.01.2024 г.).

14. Keogh E., Lin J., Fu A. HOT SAX: efficiently finding the most unusual time series subsequence. // Proceedings of the 5th IEEE International Conference on Data Mining, ICDM'05 (Houston, Texas, November, 27–30, 2005), 2005. – 8 pp.

15. Lin J., Keogh E.J., Fu A.W., Herle H.V. Approximations to magic: Finding unusual medical time series. // 18th IEEE Symposium on Computer-Based Medical Systems (CBMS 2005), 23–24 June 2005, Dublin, Ireland. IEEE Computer Society, 2005. – 329–334 pp.

16. Matplotlib. [Электронный ресурс] URL: https://pypi.org/project/matplotlib/ (дата обращения: 11.01.2024 г.).

17. Mueen A., Keogh E.J., Zhu Q., Cash S., Westover M.B. Exact Discovery of Time Series Motifs. // Proceedings of the SIAM International Conference on Data Mining, SDM 2009 (Sparks, Nevada, USA, April, 30 – May, 2, 2009). SIAM, 2009. – 473–484 pp.

18. Postgresql. [Электронный ресурс] URL: https://www.postgresql.org/ (дата обращения: 14.02.2024 г.).

19. Pydantic. [Электронный ресурс] URL: https://docs.pydantic.dev/latest/ (дата обращения: 19.02.2024 г.).

20. Real Time Intelligence Desktop. [Электронный ресурс] URL: https://github.com/real-time-intelligence/real-time-intelligence-desktop/blob/main/docs/guides/user/user-guide-ru.md (дата обращения: 05.02.2024 г.).

21. Shukla R.K., Sharma P., Samaiya N., Kherajani M. Web Usage Mining-A Study of Web data pattern detecting methodologies and its applications in

Data Mining. // 2nd International Conference on Data, Engineering and Applications (IDEA), Bhopal, India, 2020. – 1–6 pp.

22. SqlAlchemy. [Электронный ресурс] URL: https://docs.sqlalchemy.org/en/20/orm/ (дата обращения: .01.2024 г.).

23. Stumpy. [Электронный ресурс] URL: https://pypi.org/project/stumpy/ (дата обращения: 11.01.2024 г.).

24. Timescale. [Электронный ресурс] URL: https://www.timescale.com/ (дата обращения: 12.01.2024 г.).

25. Vue. [Электронный ресурс] URL: https://vuejs-doc-ru.vercel.app/guide/introduction.html (дата обращения: 22.02.2024 г.).

26. Yeh C.-C.M., Zhu Y., Ulanova L., Begum N., Ding Y., Dau H.A., Silva D.F., Mueen A., Keogh E.J. Matrix Profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. // Proc. of the IEEE 16th Int. Conf. on Data Mining, ICDM 2016, Barcelona, Spain, 12–15 December, 2016. – 1317–1322 pp.

27. Zhu Y., Gharghabi S., Silva D.F., Dau H.A., Yeh C.-C.M., Senobari N.S., Almaslukh A., Kamgar K., Zimmerman Z., Funning G., Mueen A., Keogh E. The Swiss army knife of time series data mining: Ten useful things you can do with the matrix profile and ten lines of code. // Data Min. Knowl. Discov., 2020. – 949–979 pp.

28. Zhu Y., Imamura M., Nikovski D., Keogh E., Matrix Profile VII: Time Series Chains: A New Primitive for Time Series Data Mining (Best Student Paper Award). // 2017 IEEE International Conference on Data Mining (ICDM), New Orleans, LA, USA, 2017. – 695–704 pp.

29. Zhu Y., Zimmerman Z., Senobari N.S., Yeh C.-C.M., Funning G., Mueen A., Brisk P., Keogh E.J. Matrix profile II: Exploiting a novel algorithm and GPUs to break the one hundred million barrier for time series motifs and joins. // Proc. of the IEEE 16th Int. Conf. on Data Mining, ICDM 2016, Barcelona, Spain, 12–15 December, 2016. – 739–748 pp.

30. Иванова Е.В., Цымблер М.Л. Внедрение концепции матричного профиля в реляционную СУБД для интеллектуального анализа временных рядов. // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика, 2021. – Т. 10, № 3. – С. 72–87.

31. Цымблер М.Л., Полонский В.А., Юртин А.А. Об одном методе восстановления пропущенных значений потокового временного ряда в режиме реального времени. // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика, 2021. – Т. 10, № 4. – С. 5–25.

ПРИЛОЖЕНИЯ

Приложение А. Структура таблиц базы данных

В таблицах 1–9 приведены внутренние структуры таблицы баз данных.

Таблица 1 – Структура таблицы «Users»

Наименование колонки	Тип	Описание
id	biginteger	Идентификатор пользователя
login	string	Логин пользователя
hashed_password	string	Хешированный пароль пользователя
mail	string	Почта пользователя

Таблица 2 – Структура таблицы «Projects»

Наименование колонки	Тип	Описание
id	biginteger	Идентификатор проекта
user_id	biginteger	Идентификатор пользователя
ts_ids	biginteger[]	Массив идентификаторов координат вр. ряда
dim	integer	Размерность временного ряда
name	string	Название проекта
create_date	date	Дата создания проекта
description	string	Описание проекта

Таблица 3 – Структура таблицы «Time_series»

Наименование колонки	Тип	Описание
id	biginteger	Идентификатор временного ряда
project_id	biginteger	Идентификатор проекта
name	string	Название временного ряда
stamp	timestamp[]	Массив временных меток ряда
value	float[]	Значения временного ряда
len	integer	Длина временного ряда
description	string	Описание временного ряда

Таблица 4 – Структура таблицы «Matrix_profiles»

Наименование колонки	Тип	Описание
id	biginteger	Идентификатор матричного профиля
ts_id	biginteger	Идентификатор временного ряда
nnDist	float[]	Массив расстояний до ближайших соседей
nnIdx	integer[]	Массив индексов ближайших соседей
subseqLen	integer	Длина подпоследовательности
left_nnIdx	integer[]	Массив индексов левых ближайших соседей
right_nnIdx	integer[]	Массив индексов правых ближайших соседей

Таблица 5 – Структура таблицы «Motifs»

Наименование колонки	Тип	Описание
id	biginteger	Идентификатор top-k мотивов
ts_id	biginteger	Идентификатор временного ряда
indexes_Left	integer[]	Массив индексов левых частей мотива
indexes_Right	integer[]	Массив индексов правых частей мотива
nnDist	float[]	Массив расстояний между правой и левой частями мотивов
subseqLen	integer	Длина подпоследовательности
k	integer	Количество мотивов

Таблица 6 – Структура таблицы «Discords»

Наименование колонки	Тип	Описание
id	biginteger	Идентификатор top-k диссонансов
ts_id	biginteger	Идентификатор временного ряда
indexes	integer[]	Массив индексов начала диссонансов
nnDist	float[]	Массив расстояний до ближайших соседей
subseqLen	integer	Длина подпоследовательности
k	integer	Количество диссонансов

Таблица 7 – Структура таблицы «Snippets»

Наименование колонки	Тип	Описание
id	biginteger	Идентификатор top-k сниппетов
ts_id	biginteger	Идентификатор временного ряда
indexes	integer[]	Массив индексов начала сниппетов
freq	float[]	Массив покрытий
subseqLen	integer	Длина подпоследовательности
k	integer	Количество сниппетов

Таблица 8 – Структура таблицы «Chains»

Наименование колонки	Тип	Описание
id	biginteger	Идентификатор k цепочек
ts_id	biginteger	Идентификатор временного ряда
indexes	integer[]	Массив индексов начала цепочек
numElements	integer[]	Массив в котором каждый элемент, это количество элементов в определенной цепочке, расположенных в порядке убывания
subseqLen	integer	Длина подпоследовательности
k	integer	Количество цепочек

Таблица 9 – Структура таблицы «Found_primitives»

Наименование колонки	**Тип**	**Описание**
id	biginteger	Идентификатор найденных примитивов
project_id	biginteger	Идентификатор проекта
discord_ids	biginteger[]	Массив идентификаторов диссонансов
motif_ids	biginteger[]	Массив идентификаторов мотивов
snippet_ids	biginteger[]	Массив идентификаторов сниппетов
chain_ids	biginteger[]	Массив идентификаторов цепочек
matrix_pr_ids	biginteger[]	Массив идентификаторов матричных профилей

Приложение Б. Макеты диалоговых форм приложения

На рисунках 1–2 изображение макеты дополнительных диалоговых форм приложения.

Рисунок 1 – Макет формы авторизации

Рисунок 2 – Макет формы регистрации